AN INTERNATIONAL CONSENSUS ON THE SOFTWARE ENGINEERING BODY OF KNOWLEDGE

Executive Editors: *Alain Abran*, École de technologie supérieure, *James W. Moore*, The MITRE Corp.

Editors: *Pierre Bourque*, École de technologie supérieure, *Robert Dupuis*, Université du Québec à Montréal; Project Champion: *Leonard L. Tripp*, Chair, Professional Practices Committee, IEEE Computer Society (2001-2003)

In spite of the millions of software professionals worldwide and the ubiquitous presence of software in our society, software engineering has only recently reached the status of a legitimate engineering discipline and a recognized profession.

Achieving consensus by the profession on a core body of knowledge is a key milestone in all disciplines and had been identified by the IEEE Computer Society as crucial for the evolution of software engineering towards professional status. The Guide to the Software Engineering Body of Knowledge (SWEBOK), written under the auspices of the Professional Practices Committee, is part of a multi-year project designed to reach such a consensus.

WHAT IS SOFTWARE ENGINEERING?

The IEEE Computer Society defines software engineering as:

"(1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

(2) The study of approaches as in (1)."[1]

WHAT IS A RECOGNIZED PROFESSION?

For software engineering to be fully known as a legitimate engineering discipline and a recognized profession, consensus on a core body of knowledge is imperative. This fact is well illustrated by Starr when he defines what can be considered a legitimate discipline and a recognized profession. In his Pulitzer Prize-winning book on the history of the medical profession in the USA, he states that:

"The legitimization of professional authority involves three distinctive claims: first, that the knowledge and competence of the professional have been validated by a community of his or her peers; second, that this consensually validated knowledge rests on rational, scientific grounds; and third, that the professional's judgment and advice are oriented toward a set of substantive values, such as health. These aspects of legitimacy correspond to the kinds of attributes–collegial, cognitive, and moral–usually embodied in the term "profession."[2]

WHAT ARE THE CHARACTERISTICS OF A PROFESSION ?

Gary Ford and Norman Gibbs studied several recognized professions, including medicine, law, engineering, and accounting.[3] They concluded that an engineering profession is characterized by several components:

- An initial *professional education* in a curriculum validated by society through *accreditation*
- Registration of fitness to practice via voluntary *certification* or mandatory *licensing*
- Specialized *skill development* and *continuing professional education*
- Communal support via a *professional society*
- A commitment to norms of conduct often prescribed in a *code of ethics*

This Guide contributes to the first three of these components. Articulating a Body of Knowledge is an essential step toward developing a profession because it represents a broad consensus regarding what a software engineering professional should know. Without such a consensus, no licensing examination can be validated, no curriculum can prepare an individual for an examination, and no criteria can be formulated for accrediting a curriculum. The development of consensus is also a prerequisite to the adoption of coherent skills development and continuing professional education programs in organizations.

WHAT ARE THE OBJECTIVES OF THE SWEBOK PROJECT?

The Guide should not be confused with the Body of Knowledge itself, which already exists in the published literature. The purpose of the Guide is to describe what portion of the Body of Knowledge is generally accepted,

[1] "IEEE Standard Glossary of Software Engineering Terminology," IEEE, Piscataway, NJ std 610.12-1990, 1990.

[2] P. Starr, The Social Transformation of American Medicine: Basic Books, 1982. p. 15.

[3] G. Ford and N. E. Gibbs, "*A Mature Profession of Software Engineering*," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical CMU/SEI-96-TR-004, January 1996.

to organize that portion, and to provide a topical access to it. The Guide to the Software Engineering Body of Knowledge (SWEBOK) was established with the following five objectives:

1. To promote a consistent view of software engineering worldwide
2. To clarify the place–and set the boundary–of software engineering with respect to other disciplines such as computer science, project management, computer engineering, and mathematics
3. To characterize the contents of the software engineering discipline
4. To provide a topical access to the Software Engineering Body of Knowledge
5. To provide a foundation for curriculum development and for individual certification and licensing material

The first of these objectives, a consistent worldwide view of software engineering, was supported by a development process which engaged approximately 500 reviewers from 42 countries in the Stoneman phase (1998-2001) leading to the Trial version, and over 120 reviewers from 21 countries in the Ironman phase (2003) leading to the 2004 version. More information regarding the development process can be found in the Preface and on the Web site (www.swebok.org). Professional and learned societies and public agencies involved in software engineering were officially contacted, made aware of this project, and invited to participate in the review process. Associate editors were recruited from North America, the Pacific Rim, and Europe. Presentations on the project were made at various international venues and more are scheduled for the upcoming year.

The second of the objectives, the desire to set a boundary for software engineering, motivates the fundamental organization of the Guide. The material that is recognized as being within this discipline is organized into the first ten Knowledge Areas (KAs) listed in Table 1. Each of these KAs is treated as a chapter in this Guide.

Table 1 The SWEBOK Knowledge Areas (KAs).

Software requirements
Software design
Software construction
Software testing
Software maintenance
Software configuration management
Software engineering management
Software engineering process
Software engineering tools and methods
Software quality

In establishing a boundary, it is also important to identify what disciplines share that boundary, and often a common intersection, with software engineering. To this end, the Guide also recognizes eight related disciplines, listed in Table 2. Software engineers should, of course, have knowledge of material from these fields (and the KA descriptions may make reference to them). It is not, however, an objective of the SWEBOK Guide to characterize the knowledge of the related disciplines, but rather what knowledge is viewed as specific to software engineering.

Table 2 Related disciplines.

◆	Computer engineering	◆	Project management
◆	Computer science	◆	Quality management
◆	Management	◆	Software ergonomics
◆	Mathematics	◆	Systems engineering

HIERARCHICAL ORGANIZATION

The organization of the KA descriptions or chapters supports the third of the project's objectives – a characterization of the contents of software engineering. The Guide uses a hierarchical organization to decompose each KA into a set of topics with recognizable labels. A two- or three-level breakdown provides a reasonable way to find topics of interest. The Guide treats the selected topics in a manner compatible with major schools of thought and with breakdowns generally found in industry and in software engineering literature and standards. The breakdowns of topics do not presume particular application domains, business uses, management philosophies, development methods, and so forth. The extent of each topic's description is only that needed to understand the generally accepted nature of the topics and for the reader to successfully find reference material. After all, the Body of Knowledge is found in the reference material themselves, and not in the Guide.

REFERENCE MATERIAL AND MATRIX

To provide a topical access to the knowledge–the fourth of the project's objectives–the Guide identifies reference material for each KA, including book chapters, refereed papers, or other recognized sources of authoritative information. Each KA description also includes a matrix relating the reference material to the listed topics. The total volume of cited literature is intended to be suitable for mastery through the completion of an undergraduate education plus four years of experience.

In this edition of the Guide, all KAs were allocated around 500 pages of reference material, and this was the specification the associate editors were invited to apply. It may be argued that some KAs, such as software design for instance, deserve more pages of reference material than others. Such modulation may be applied in future editions of the Guide.

It should be noted that the Guide does not attempt to be comprehensive in its citations. Much material that is both suitable and excellent is not referenced. Material was selected in part because–taken as a collection–it provides coverage of the topics described.

Depth of Treatment

From the outset, the question arose as to the depth of treatment the Guide should provide. The project team adopted an approach which supports the fifth of the project's objectives–providing a foundation for curriculum development, certification, and licensing. The editorial team applied the criterion of *generally accepted* knowledge, to be distinguished from advanced and research knowledge (on the grounds of maturity) and from specialized knowledge (on the grounds of generality of application). The definition comes from the Project Management Institute: "The generally accepted knowledge applies to most projects most of the time, and widespread consensus validates its value and effectiveness".[4]

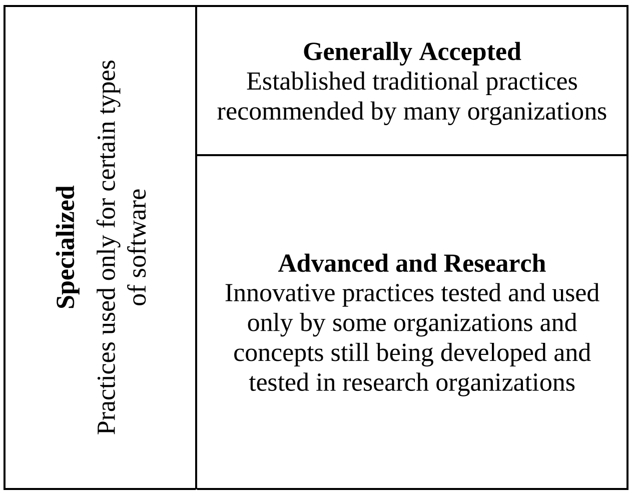
Specialized
Practices used only for certain types of software
Generally Accepted
Established traditional practices recommended by many organizations
Advanced and Research
Innovative practices tested and used only by some organizations and concepts still being developed and tested in research organizations

Figure 1 Categories of knowledge

However, the term "generally accepted" does not imply that the designated knowledge should be uniformly applied to all software engineering endeavors–each project's needs determine that–but it does imply that competent, capable software engineers should be equipped with this knowledge for potential application. More precisely, generally accepted knowledge should be included in the study material for the software engineering licensing examination that graduates would take after gaining four years of work experience. Although this criterion is specific to the U.S. style of education and does not necessarily apply to other countries, we deem it useful. However, the two definitions of generally accepted knowledge should be seen as complementary.

The Knowledge Areas

Figure 1 maps out the eleven chapters and the important topics incorporated within them. The first five KAs are presented in traditional waterfall life cycle sequence. However, this does not imply that the Guide adopts or encourages the waterfall model, or any other model. The subsequent KAs are presented in alphabetical order, and those of the related disciplines are presented in the last chapter. This is identical to the sequence in which they are presented in this Guide.

Acknowledgements: This text is an extract of the 'Introduction' to the 'Guide of the Software Engineering Body of Knowledge' published by the IEEE Computer Society which owns the copyrights as stated as follows:

[4] *A Guide to the Project Management Body of Knowledge,* 2000 Edition, Project Management Institute, Newport Square, PA. www.pmi.org.

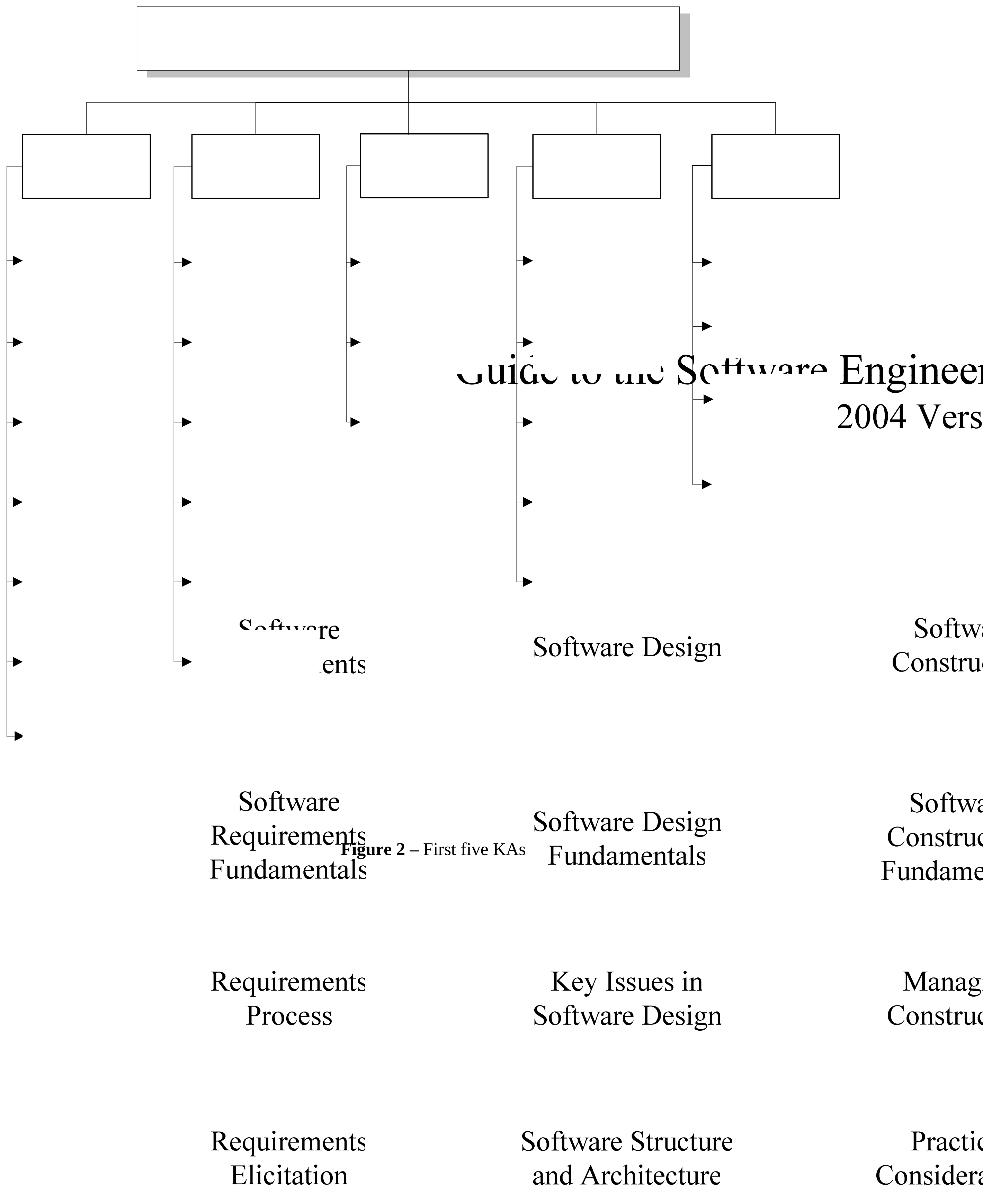

Figure 2 – First five KAs

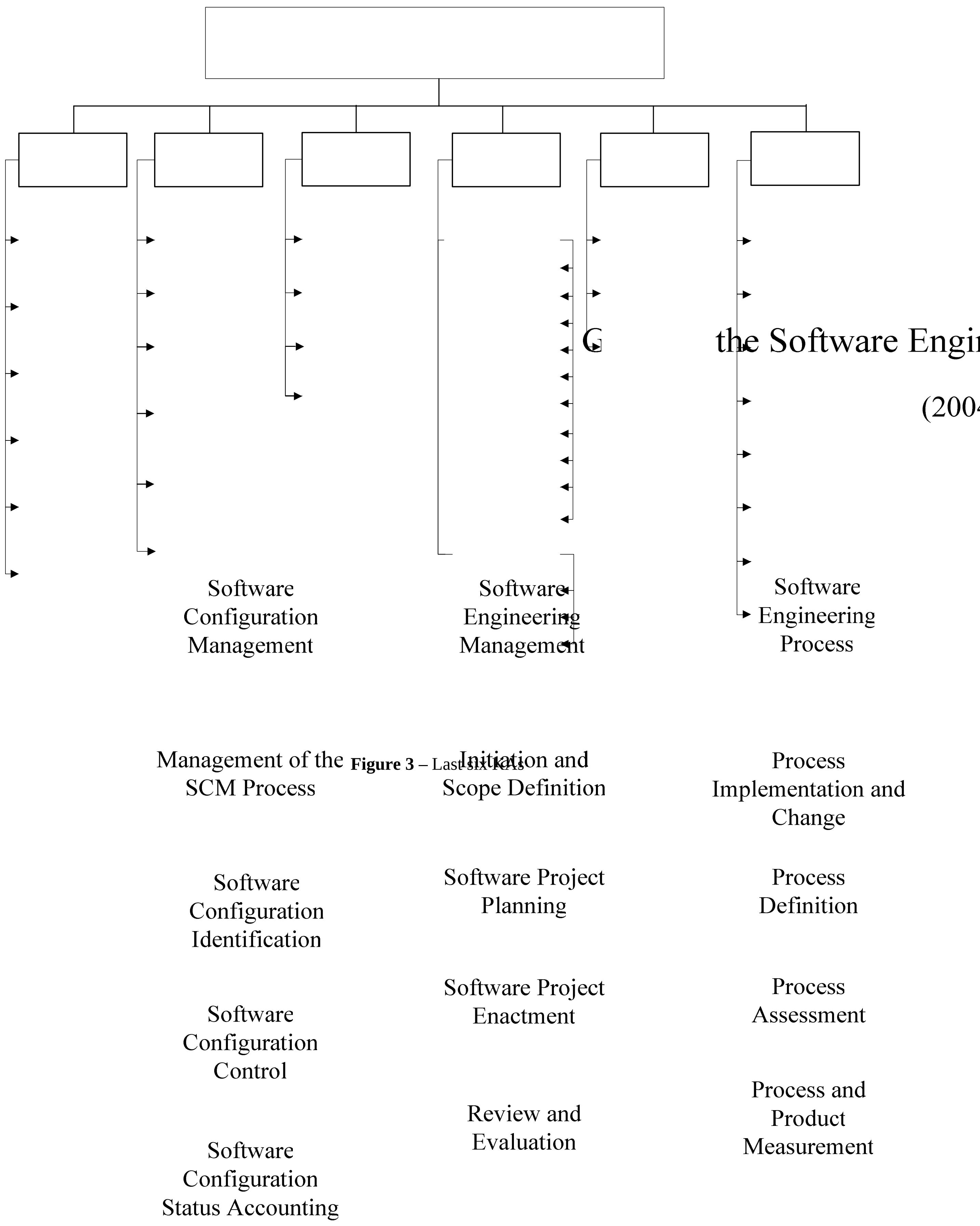
G
the Software Engi
(200
Software Configuration Management
Software Engineering Management
Software Engineering Process
Management of the SCM Process
Initiation and Scope Definition
Process Implementation and Change
Software Configuration Identification
Software Project Planning
Process Definition
Software Configuration Control
Software Project Enactment
Process Assessment
Review and Evaluation
Process and Product Measurement
Software Configuration Status Accounting

Figure 3 – Last six KAs